LOIS - Lowell Observatory Instrumentation System: A modular control system for astronomical instrumentation.

Brian W. Taylor[a], Edward W. Dunham[a],Adam J. Gould [b], David J. Osip[b], James L. Elliot[b]

[a]Lowell Observatory, 1400 West Mars Hill Road, Flagstaff, AZ., USA
[b]Mass. Institute of Technology, 54-420 MIT, Cambridge, MA., USA

ABSTRACT

The Lowell Observatory Instrumentation System (LOIS) is the control system for a series of new instruments at Lowell, including the SOFIA* first light instrument, HOPI† . Since these instruments will incorporate various detector systems and will be used with several telescopes, the concept of a loadable modular based design was developed. The fundamental idea is to view the telescope, camera, and other instrument components as separate, interchangeable entities.

This interchangeable module system is based on the scripting language Tcl/Tk, using the extensibility of the language to change the hardware functionality of a command without changing the command. This provides a means to structure a set of static commands for different hardware systems. Using the same series of scripts and commands, HOPI and other instruments can maintain a library of observation programs to be utilized at different observatory sites. LOIS also provides an interface for control by a high-level external program to implement semi-autonomous or robotic system operation.

LOIS provides the flexibility to add new instrument, telescope, and camera interfaces without changing any other aspect of the main program, analysis, or image display functions, and provides a static and consistent interface for observers using the different camera systems and telescopes. It is already operating several CCD cameras at Lowell, another system at NASA Ames in support of the Kepler mission technology demonstration, and is being used to test the SOFIA mission control software system's instrument interface.

Keywords: CCD, imaging, software, Tcl/Tk, SOFIA

1. INTRODUCTION

The LOIS development effort was prompted by the need for an instrument control system to be flexible enough to handle a multitude of different combinations of instruments, telescopes and cameras while providing a consistent and stable interface for observers. In order to accommodate this need for both current and future instrumentation a modular paradigm was implemented at the onset of development. Within this paradigm, each hardware system is considered an independent entity with its own set of functions and commands that can be loaded dynamically into the LOIS core. We maintain a consistent interface for observers by grouping hardware systems based on function with a main set of commands defined for each group, complemented by any necessary specialized commands for each hardware function that is outside the general definition of that group. The modular ideal is not confined to the hardware systems, but is extended to functions for analysis, image storage, display, etc. All commands are implemented so that a command can have a different function depending on which module is loaded. For example, a store command could store an image file as a FITS, JPG, GIF, or raw binary depending on which storage module the user chose to load into the LOIS core.

Further author information: (Send correspondence to Brian W. Taylor)
B.W.T.: E-mail: taylor@lowell.edu
E.W.D.: E-mail: dunham@lowell.edu
A.J.G.: E-mail: agould@mit.edu
D.J.O.: E-mail: osip@astron.mit.edu
J.L.E.: E-mail: jle@mit.edu

*Stratospheric Observatory For Infrared Astronomy
†High-Speed Occultation Photometer and Imager

Although the LOIS development is still in its infancy it is currently operating four production line CCD cameras including the Kepler[1,2] Mission Test CCD, five telescopes (includes the SOFIA MCS[‡] Test Simulation), and various instrument components. In the following sections we provide the reasoning and requirements for the choices that were made during the initial planning stages of development, a description of the internal architecture, and the current status and future plans for LOIS.

2. APPLICATIONS AND REQUIREMENTS

This section describes the planned applications for LOIS (Section 2.1) and the requirements that fall out of these applications (Section 2.2). LOIS is required to support a wide variety of instrument operating modes, so we briefly describe these modes in Section 2.3. Finally we lay out a number of basic implementation decisions in Section 2.4 before moving on the architecture discussion of Section 3.

2.1. INTENDED USES:

The data system design is partially defined by the requirement that it be sufficiently modular that it can, with relatively minor module replacement, operate the following instruments:

1. HOPI, at Lowell with the Move[§] telescope control system, on SOFIA, and with Magellan.
2. The Harris/Palomar[3] CCD systems using a PC running LINUX as the CCD controller with the Move system
3. The Kepler CCD system for the Kepler Mission Test Experiment.
4. The LONEOS[¶] 4K x 4K mosaic camera.
5. Four CCD imagers to be based on Lick/Loral 2Kx2K CCDs and SDSU controller[4] with the Move system.
6. The White spectrograph and its associated control systems, with the Move system
7. An MIT instrument for use with Magellan and with the Move system
8. The Lowell/BU IR camera/spectrograph, Mimir, with the Move system
9. The SITe/OSU 2K CCD system using a PC running LINUX as the CCD controller with the Move system

2.2. REQUIREMENTS

In this section we define the requirements we have placed on LOIS. These are broken into two main groups, the major functional requirements and specific, detailed requirements. Many of these requirements are due to the intended use of HOPI for initial performance testing of SOFIA[5] and to its use for occultation observations. These applications place demands on a CCD system that are quite unusual.

2.2.1. KEY FUNCTIONAL REQUIREMENTS:

- User Interface:
  - Graphical user interface suitable for infrequent users
  - Command line and scriptable/macro interface for power users
  - Port for accepting command streams from other programs
  - Ability to control and receive data from a remote LOIS process
  - All appropriate operations possible with any user interface.

[‡]**M**ission **C**ontrol **S**oftware
[§]Lowell Observatory Telescope motion control system
[¶]**L**owell **O**bservatory **N**ear **E**arth **O**bject **S**earch

- Telescope Interfaces:
  - Move
  - SOFIA
  - Magellan
  - simple guide/focus interfaces.
- Speed: Take and store data locally at a sustained rate of 8 Mbytes/sec.
- Operating Modes[‖]:
  - CCD: find, single frames, series, high speed series, strips, dots.
  - IR: find, single frames, high speed series, multiple non-destructive readouts[**].
  - Binning available in all CCD systems.
  - Data playback capability available in all systems.
  - Autoguiding included in some systems.
  - Simple self-test for verification after instrument change.
- In-Program Data Analysis:
  - Optional automatic flat/dark/bias calibration with bad pixel masking.
  - All other analysis functions take place on a single frame
  - More sophisticated analyses carried out in external programs
  - Allow real-time access to in-memory data by external programs.
  - Allow analysis on old images while new ones are being taken.
- Data Format: FITS 2d or 3d time series files, with one file per subframe if multiple subframes are supported.
- Expandability:
  - Allow for future implementation of adaptive optics
  - Plan for eventual fully automatic instrument/telescope operation.

2.2.2. Specific Requirements

- Controls:
  - Take automatically or manually stored data, pause, abort, all based on current operating mode.
  - Color Tables, autoscaling options, and mapping options.
  - Subframe definition, using either mouse or keyboard with up to ten subframes supported.
  - Integration and other timing definition, with synchronization to UTC if needed.
  - Data storage options: Store to disk, store to tape; send to another LOIS process, or any combination.
  - Instrument specific control will be customized for each instrument in the areas of hardware control, communication, and user interface.
  - Scripting GUI able to create, store, load, and edit scripts. Scripts in plain text.
  - Special functions for scripts: wait till time, wait for time interval, loops, branch on analysis results.

[‖] see Sect. 2.3 for mode descriptions

[**] Also known as Fowler sampling

- Analysis Modes:
  - Marginal analysis: Min/max/average/median, FWHM, position.
  - Individual pixel values and area pixel dump using the mouse.
  - Circular aperture photometry, row & column plots, radial plots, histogram of a rectangular region.
  - Quick and dirty spectral extraction for spectrographs.
  - Time series software strip chart.
- Telescope Control:
  - Focus control including automatic focus algorithms and filter dependent focus settings.
  - Field centering.
  - Allow pass through of all commands to the telescope system.
  - Record telescope status for inclusion in data file headers
- Display:
  - Image display with zoom/pan, and overlays for subframes and analysis regions.
  - Detector system status - subframes, timing, operating mode.
  - Instrument status - filters, focus, grating tilt, shutter state, viewers, etc.
  - Telescope status: RA/Dec, epoch, track rates, focus, etc.

2.3. INSTRUMENT OPERATING MODES

The variety of instruments to be operated by LOIS and the unusual applications of some of these instruments requires LOIS to support many disparate observing modes. Many of these modes were developed earlier for similar purposes and additional descriptions may be found elsewhere[6,7]. For convenience, we give short descriptions here of the modes we expect LOIS to support.

- **Find:** Continuous readout, with or without shuttering, for finding and manually centering an object or starfield on the detector.
- **Single Frames:** Simple single frames using a shutter.
- **Series:** A series of single frames with precisely defined exposure times and intervals between exposures. Normally the exposures will be shuttered.
- **High speed series:** Use frame transfer mode to allow unshuttered high-speed CCD readout with precisely defined exposure times and intervals between exposures. In IR systems this mode is implemented by successive reads without intervening detector clear operations. This is the normal occultation observing mode.
- **Strips:** Read individual rows with a precisely defined interval with shutter open. This is often called Time Delay and Integrate(TDI) mode, and can also be used with binning for high speed data acquisition.
- **Dots:** Open the shutter, integrate, shift the charge image by a set number of rows, integrate, shift, etc. Finally the shutter is closed and the CCD is read out. Integrations as short as tens of microseconds can be obtained in this mode.
- **Autoguiding:** In a two-CCD or frame transfer system, use one CCD or one half of the CCD to take data while the other is read relatively rapidly to provide images for deriving telescope guiding corrections.
- **Noise:** Take pairs of images, subtract them, and find the rms difference divided by the square root of two, the average difference, and the mean brightness. This is for determining the noise of any system and the gain of CCD systems. It is convenient to run this mode continuously when locating noise sources.

- **Dither:** This CCD submode involves clocking the charge both ways in the parallel direction during integration. Depending on the distance the charge moves, this can be used for dark current reduction, antiblooming, sky subtraction, and trap finding in the lab.
- **Playback:** This is not a data acquisition mode, but instead allows data already obtained to be played back to allow simple analysis to be done.
- **Multiple Readout:** In IR systems it is possible to read the detector nondestructively. The Reset-Read-Read scheme (Fowler sampling) is essentially the same as a CCD double correlated sample. The array is reset, then read prior to and after integration and the difference between the two reads is the signal. Multiple reads either after the integration is complete or during the integration serve to reduce the read noise.

2.4. KEY IMPLEMENTATION DECISIONS:

We have decided to make use of external software packages to the greatest degree possible in order to reduce our development effort. This plan is not without risk, particularly in the area of data acquisition speed. For example, image display is often too slow, so we display only as many images as we can when operating at high data rate.

- Scripting will use the built-in Tcl/Tk interpreter, providing loops, branching, and variable substution.
- Image display will use Ximtool[8] and SAOtng,[9] modified as required to conform to our module interface standard. Other display interfaces that conform to the LOIS internal interface may be written but will not be part of LOIS.
- FITS I/O will use the *cfitsio*[10] library available from GSFC at http://heasarc.gsfc.nasa.gov/fitsio/.
- Most CCD systems will use Leach/SDSU controllers. The IR system will also use Leach/IR Labs electronics. The Harris CCD system will use a single PC running Linux to feed data to LOIS across the network. The Atwood system for the 2K SITe CCD be similar to the Harris system.
- Communication with the telescope system will be via RS-232 or ethernet as appropriate.
- Communication with the CCD controller will be via bus and device driver for the Leach systems and via TCP/IP sockets for the PC systems.
- Communication with instrument-specific controls will be via RS-232 or ethernet as appropriate.
- Detector readout, data storage, image display, and automatic flat/dark/bias calibration will occur simultaneously.

3. ARCHITECTURE

3.1. Tcl/Tk Interface

LOIS is based on the Tcl/Tk interpretive scripting language. Tcl is a full featured language that provides an avenue for extending the scripting language with new commands. This, coupled with the Tk graphics component makes the most sensible choice for our small development team. By adopting Tcl/Tk as the foundation for LOIS, we solved many of the basic scripting and GUI requirements that were defined at the onset of development.

3.1.1. Scripting

Being able to script in an observing session was one of the main requests from the astronomers we polled as we developed the requirements for LOIS. When running long standardized observing programs, scripting makes the process much easier and increases the number of objects that can be observed in a session. In fact, an observing session can be programmed completely before observing or can be derived from a previous session and then executed in a robotic mode with the observer in place for intervention during unforeseen events. Efficiency is an absolute must for SOFIA. In order to make the most productive use of the flight time available, scripts will be written ahead of time utilizing simulators to plan the flight and build the observing scripts for individual flight legs or even the entire flight.

Choosing Tcl/Tk for the extensibility of the language provided the required scripting capabilities including all of the branching, loops, variable substution and condition checking that one would expect from any scripting language. In addition to these functions, Tcl/Tk provides the ability for the observer to add graphics and network socket functions to their own scripts increasing the feedback for the observing session.

3.1.2. Module or Package Templates

Using the C API[††] in the Tcl/Tk distribution, modules are created as Tcl/Tk packages and loaded as needed during an observing session. The module template forms the basis for the command grouping of like hardware functions. The template consists of a Tcl/Tk wrapper to parse and interpret commands, execution threads for synchronous and asynchronous hardware functions, and a series of function pointers to the hardware function set. It is these functions that do the customization of each hardware module minimizing the amount of new software code that will be needed in order to prototype a new module. In our case we are using ANSI C for our module template and hardware functions but there is no limitation on what language could be used.

3.2. Core and Inter-Module Communication

Inter-module communication becomes a problem since the modules are designed to act as independent entities. In order to be able to call functions from one module to be executed by another module, some path must be provided to the functions called. Also since LOIS is required to run commands both in scripts and asynchronously, two paths of execution are required in order to execute a command while a script was running. In most cases a series of synchronous and asynchronous commands in a script can be handled by a single Tcl Interpreter, but while this script is running that interpreter is blocked and no more commands can be executed until the script is done. This causes a problem with graphical update commands and with the commands that are called from inside a module to be executed by another module, a focus move sent to a telescope for example. In order to be able to continually update the GUI and execute inter-module commands, we arrived at a solution of threads and dual Tcl interpreters, relying heavily on the IPC[‡‡] available in the POSIX 1b real time standard. By using message queues and shared memory for communication between the LOIS core and the modules loaded we are able to surmount the problems in communication while maintaining module independence.

3.3. Threads and Dual Tcl Interpreters

LOIS uses threads throughout the whole software package in order to meet the asynchronous requirement of the design. Although threads are a perfect way to provide asynchronous command execution, there are some drawbacks to using them. Threads have a small performance cost when used on a single CPU machine and they increase the difficulty in maintaining code. The programmer must be extremely careful about protecting critical sections of code with mutex locks so that important memory sections are not executed out of order. But with the number of asynchronous functions and amount of memory that is maintained in the process, using a fork approach would be far too costly in memory to use.

LOIS utilizes POSIX threads or pthreads[11] in both its core and modules. Upon start up LOIS initializes two threads and in each of those threads initializes a Tcl interpreter. Keeping the interpreters separated in the threads accomplishes two major functions. It protects the memory and variable use in the interpreters and provides a method to execute graphical interface commands while executing commands in a script. By default all graphical commands are executed on the first interpreter or "graphical" interpreter while module commands are executed on the second "command" interpreter. To protect the variable space that the user must interact with, all variables are defined and manipulated inside the graphical interpreter. Since threads share the memory space of the process it is safe to read the variables. But, in order to protect the memory space, modules manipulate variables by sending graphical commands to the graphics interpreters via the message queues.

[††] **A**pplication **P**rogram **I**nterface
[‡‡] **I**nter **P**rocess **C**communication

3.3.1. Command Execution, Scheduling, and Results

Command execution and results are handled by message queues in order to provide a pathway into the LOIS core and out to the modules. These queues provide a means to have a multi point non-determinate method of communication. In the command execution domain the queue allows commands to come from several modules at the same time. The queue has a means to assign priority to commands that need to be executed before any other commands in the queue, for example aborting an exposure or telescope move.

Commands are entered into the command queue as a structure containing priority, interval between repeated command execution, number of times to repeat the command, a time to execute the command, and the command itself. This structure is handled by a scheduler that uses the structure variables to determine when and how many times a command should be executed and assigns a process tag to the command.

After a command is executed, its results are returned by another message queue to the module that sent the command. When the results of a command are returned by the Tcl interpreter they are placed in the results queue tagged with the process tag to be read by the module that sent the command. The module has the ability to block execution while waiting for the command to return a result or it can continue execution asynchronously only interrupting if some error occurs.

3.3.2. Shared Memory and Pipes

LOIS relies on shared memory to communicate between threads inside of the module and between modules. Shared memory is used to store the structures that describe the current state of the running process and the commands that are currently being executed. Shared memory is also used as the CCD image buffer for the CCD readout. If a module uses the image buffer in a non-destructive mode, it can have access to the buffer. This feature allows analysis without unnecessary image copying.

3.3.3. Synchronous and Asynchronous Command Execution

Being able to handle both synchronous and asynchronous command execution was decided upon in order to improve the efficiency of the observing session. Utilizing the independence of different hardware controllers to make the commands execute simultaneously reduces the delay between exposures. This is of course dependent on the hardware involved and how the module has been defined. If the hardware is capable of accepting and acting on commands in parallel then the module can be designed with the ability to send those commands asynchronously. LOIS can then continue on executing the script or wait on either or both of the commands to finish execution. This feature is heavily used in the LONEOS camera to reduce the dead time between exposures to only 13 seconds including the 10.5 second CCD readout time.

4. CONCLUSION

4.1. Current Status

4.1.1. Version 1.0.1

Currently the first revision of the first release of LOIS has been running for about one year at the Lowell Observatory Hall and Perkins Telescopes using the Harris/Palomar CCD controllers. This version of LOIS is limited in functionality in that it does not contain any scripting abilities and has very limited asynchronous execution.

4.1.2. Version 1.1 Alpha

An operational version of the new LOIS core is running the Kepler Mission Test CCD. This version also has limited scripting but the asynchronous command capabilities are starting to be implemented in full. The stability of this alpha version was demonstrated in some long period sessions that lasted for 8 days taking 1K x 1K images continuously every 3 seconds. The Kepler Mission Test is running on a Sun Microsystems Sparc 5 with Solaris 2.6 operating system. Due to the demands on the resources of the system, POSIX 1b real time scheduling and memory locking had to be in place in order make this configuration functional.

4.1.3. Version 1.1 Beta

The first full scripting and asynchronous LOIS version is currently operating the LONEOS camera at Lowell Observatory. This version has been shown to be stable on the scale of one night and has executed a single script for a 10 hour period utilizing both telescope moves and CCD imaging. It runs on a Sun Microsystems Ultra 1 Model 140/Creator 3D with Solaris 2.7, and does not require either real time scheduling or memory locking to be used.

4.2. Future Work

The immediate next step in LOIS development will be to support the new PCI interface card for the Leach/SDSU CCD controller system. This interface will be used in all of the instruments we develop for the foreseeable future. The first of these instruments will be three Loral 2Kx2K CCD systems. These systems will run on Sparc Ultra 5 and 10 machines under Solaris 2.7. Two of the systems will only use simple single frame mode, but the third will require the Find and Strip-scanning (TDI) operations with accurate UTC synchronization as well. Implementation of additional modes will continue with the development of HOPI and Mimir over the next 2-3 years.

To date, no analysis functions have been implemented in LOIS. These functions will first be added as part of the HOPI development, and will immediately be available to all of our previous systems.

ACKNOWLEDGMENTS

This work was supported under USRA Grant 8500-98-003, and NASA Grants NAG2-1328 and NAG5-7953.

REFERENCES

1. Koch, D.G., W. Borucki, E. Dunham, J. Jenkins, L. Webster, F. Witteborn, "CCD photometry tests for a mession to detect Earth-size planets in extended soalr neighborhood", *Proc. SPIE* **4013**, (4013-75, these conferences)(2000)
2. Jenkins, J.M., F. Witteborn, D.G. Koch, E.W.Dunham, W.J. Borucki, T.F. Updike, M.A. Skinner, and S.P. Jordan, "Processing CCD images to detect transits of Earth-sized planets: Maximizing sensitivity while achieving reasonable downlink requirements", *Proc. SPIE* **4013**, (4013-76, these conferences)(2000)
3. Gunn, J.E., E.B. Emory, F.H. Harris, and J.B. Oke, "The Palomar Observatory CCD Camera", *PASP*, Vol. 99, No 616, 519-534 (June 1987)
4. Leach, R.W., F.L. Beale, and J.E. Eriksen, "New-generation CCD controller requirements and an example: The San Diego State University generation II controller", *Proc. SPIE* **3355**, 512-519 (1998)
5. Dunham, E.W., James L. Elliot, and Brian W. Taylor, "HOPI - A High-speed Occultation Photometer and Imager for SOFIA", *Proc. SPIE* **4014**, (4014-9, these conferences)(2000)
6. Dunham, E.W., R.L. Baron, J.L. Elliot, J.V. Vallerga, J.P. Doty, and G.R. Ricker, "A High Speed, Dual-CCD Imaging Photometer", *Pub. A.S.P.*, **97**, 1196-1204(1985)
7. Dunham, E.W., "Optical Instrumentation for Airborne Astronomy", in "Airborn Astronomy Symposium on the Galatic Ecosystem, M.R. Haas, J.A. Davidson, and E.F. Erickson eds., *ASP Conference Series* **73**, 517-522 (1995)
8. Tody, D. and M. Fitzpatrick, "X11/GUI development utilities and applications", IRAF Project, National Optical Astronomy Observatories (1994)
9. Mandel, E. , "The SAOtng Programming Interface", in "Astronomical Data Analysis Software and Systems VI", G. Hunt and H.E. Payne eds., *ASP Confernce Series* **125** (1997)
10. Pence, W., "CFITSIO, V2.0: A New Full-Featured Data Interface", in "Astronomical Data Analysis Software and Systems VIII", D. Mehringer, R. Plante, and D. Roberts eds.,*ASP Confernce Series* **172**, 487-489 (1999)
11. Nichols, B., D. Buttlar, and J.P. Farrell, "Pthreads Programming", O'Reilly & Associates, Inc., 101 Morris Street, Sebastopol, Ca 95472 (1996)